UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

| APPLICATION NO. | FILING DATE | FIRST NAMED INVENTOR | ATTORNEY DOCKET NO. | CONFIRMATION NO. |
|---|---|---|---|---|
| 10/620,078 | 07/15/2003 | Bowen Alpern | YOR920020352US1 (16088) | 7092 |

23389 7590 06/12/2007

SCULLY SCOTT MURPHY & PRESSER, PC
400 GARDEN CITY PLAZA
SUITE 300
GARDEN CITY, NY 11530

| EXAMINER | |
|---|---|
| CHOU, ANDREW Y | |

| ART UNIT | PAPER NUMBER |
|---|---|
| 2192 | |

| MAIL DATE | DELIVERY MODE |
|---|---|
| 06/12/2007 | PAPER |

**Please find below and/or attached an Office communication concerning this application or proceeding.**

The time period for reply, if any, is set in the attached communication.

| Office Action Summary | Application No. | Applicant(s) | |
|---|---|---|---|
| | 10/620,078 | ALPERN ET AL. | |
| | Examiner | Art Unit | |
| | Andrew Y. Chou | 2192 | |

*-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --*

**Period for Reply**

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE 3 MONTH(S) OR THIRTY (30) DAYS, WHICHEVER IS LONGER, FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

**Status**

1)☒ Responsive to communication(s) filed on *27 March 2007*.

2a)☒ This action is **FINAL**. 2b)☐ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

**Disposition of Claims**

4)☒ Claim(s) *1-30* is/are pending in the application.

4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)☐ Claim(s) _____ is/are allowed.

6)☒ Claim(s) *1-30* is/are rejected.

7)☐ Claim(s) _____ is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

**Application Papers**

9)☐ The specification is objected to by the Examiner.

10)☒ The drawing(s) filed on _____ is/are: a)☒ accepted or b)☐ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

11)☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

**Priority under 35 U.S.C. § 119**

12)☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☐ All b)☐ Some * c)☐ None of:

1.☐ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____.

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

**Attachment(s)**

1) ☒ Notice of References Cited (PTO-892)
2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)
3) ☐ Information Disclosure Statement(s) (PTO/SB/08) Paper No(s)/Mail Date _____.
4) ☐ Interview Summary (PTO-413) Paper No(s)/Mail Date. _____.
5) ☐ Notice of Informal Patent Application
6) ☐ Other: _____.

## DETAILED ACTION

1. This office action is in response to the amendment filed on 03/27/2007

2. Claims 1, 20, and 30 have been amended.

3. Claims 1-30 are pending.

### *Response to Arguments*

4. Applicant's arguments with respect to claims rejection have been considered but are moot in view of the new grounds of rejection. See Chan US 2004/0255273 A1 made of record below.

### *Claim Rejections - 35 USC § 103*

5. The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

> (a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

Claims 1-30 are rejected under 35 U.S.C. 103(a) as being unpatentable over Saxe et al. US 6,477,702 (hereinafter Saxe) in view of Chan US 2004/0255273 A1 (hereinafter Chan).

**Claim 1:**

Saxe disclose a method for analyzing software code (see for example FIG. 1, and related text) comprising the steps of:

a) automatically generating program graphs representing said code utilizing static analysis techniques (see for example column 6, lines 38-44, FIGs. 1 & 13, and related text);

c) automatically identifying potential software problems from rules set analysis results (see for example column 6, lines 61-65, FIG. 2, and related text); and,

d) reporting said software problems where one or more of best practices violations and coding errors may occur control and data flow analysis (see for example column 6, lines 61-67, FIG. 2, step 205, and related text).

Although Saxe discloses a method for automatically applying a set o f rules to said program flow analysis graphs (see for example column 6, lines 53-60, FIG. 2, and related text), Saxe fails to disclose a method for analyzing software code where:

b) automatically applying a set of rules to said program flow analysis graphs, including at least performing a reachability analysis for at least removing one or more edges to reduce reachability.

However, Chan in the same analogous art of program optimization discloses a method for analyzing software code where:

b) automatically applying a set of rules to said program flow analysis graphs, including at least performing a reachability analysis for at least removing one or more edges to reduce reachability (see for example, FIG. 4, blocks 404, 408, and associated text, e.g. page 4, [0057], lines 1-15).

Therefore, at the time the invention was made it would have been obvious to a person of ordinary skill in the art to modify the teachings of Saxe to include a method as

taught above in Chan so as to optimize the program since fewer resources are used in memory (see for example Chan, page 4, [0057]).

**Claim 2:**

Saxe further discloses the method according to Claim 1, wherein said rules set represents one or more selected from the group comprising: use of best practices and common coding errors, or combinations thereof (see for example column 7, lines 27-45, FIG. 2, item 220, and related text).

**Claim 3:**

Saxe further discloses the method according to Claim 1, wherein said reporting d) includes presenting the results in the context of corresponding source code or object code (see for example FIG. 2, item 205, 209, 210, and related text).

**Claim 4:**

Saxe further discloses the method according to Claim 1, wherein step b) includes performing rule searches applied to said program graphs (see for example column 7, lines 1-18).

**Claim 5:**

Saxe further discloses the method according to Claim 1, wherein said software code subject to said static analysis techniques comprises one or more selected from the group comprising: object code, source code, a compiler intermediate representation, of said software code, and other program representations, or combinations thereof (see for example FIG. 1, item 110, and related text).

**Claim 6:**

Saxe further discloses the method according to Claim 3, wherein a program graph includes a control analysis graph, said static analysis technique automatically generating said control analysis graphs from said software code (see for example FIGs 7 & 8, and related text).

**Claim 7:**

Saxe further discloses the method according to Claim 3, wherein a program graph includes a data flow analysis graph, said static analysis technique automatically generating said data flow analysis graph from said software code (see for example FIGs 7 & 8, and related text).

**Claim 8:**

Saxe further discloses the method according to Claim 3, wherein a program graph includes an intraprocedural control graph (see for example column 10, lines 30-50), said static analysis technique automatically generating said intraprocedural control graphs from said software code.

**Claim 9:**

Saxe further discloses the method according to Claim 3, wherein a program graph includes an interprocedural control graphs, said static analysis technique includes automatically generating said interprocedural control graphs from said software code (see for example column 10, lines 30-50).

**Claim 10:**

Saxe further discloses the method according to Claim 5 wherein said static code analysis further includes automatically identifying classes, fields, methods and class

attributes, said set of rules being further applied to said classes and class attributes (see for example FIG. 2, item 202, and related text).

**Claim 11:**

Saxe further discloses the method according to Claim 5 wherein said static code analysis further includes automatically identifying attributes of classes, methods, fields, and aspects of a programs body (see for example FIG. 2, item 202, and related text).

**Claim 12:**

Saxe further discloses the method according to Claim 5, wherein said step b) further includes the step of: receiving said program graphs and class attributes information and performing a graph rewriting technique (see for example FIG. 13, step 1304, and related text).

**Claim 13:**

Saxe further discloses the method according to Claim 12, wherein a result of applying graph rewriting includes generating a run-time characteristics model for said program (see for example column 7, lines 27-37).

**Claim 14:**

Saxe further discloses the method according to Claim 12, wherein said step b) further includes the step of receiving said program graphs and attributes information, and performing a reachability analysis (see for example column 11, lines 10-17).

**Claim 15:**

Saxe further discloses the method according to Claim 14, wherein said reachability analysis is performed with or without constraints (see for example column 11, lines 10-17).

**Claim 16:**

Saxe further discloses the method according to Claim 14, further comprising the step of employing a rule search engine (see for example FIG. 2, items 204,220, and related text) to automatically apply a set of rules (see for example FIG. 2, item 220, and related text) to said rewrite graph results, teachability analysis results and attributes to identify one or more selected from the group of: possible performance errors or problems concerning correctness, security, privacy and maintainability of said software code (see for example column 6, lines 5-21).

**Claim 17:**

Saxe further discloses the method according to Claim 14, wherein said rewrite graph technique includes traversing a program graph to locate nodes containing attributes of interest and to locate edges to add or remove from said program graph (see for example column 11, lines 52-60, FIG. 12, and related text).

**Claim 18:**

Saxe further discloses the method according to Claim 17, wherein said reachability analysis includes traversing the program graphs and adding or removing edges to extend or reduce reachability, respectively (see for example column 11, lines 52-60).

**Claim 19:**

Saxe further discloses the method according to Claim 18, wherein a rule is applied to determine whether a node representing a particular method is reachable by traversing said graph from a particular head node, said heåd node being user selectable (see for example FIG. 13, and related text).

**Claim 20:**

Saxe discloses a static analysis framework for analyzing software code said framework comprising:

rule search engine for automatically applying a set of rules to said program graphs (see for example column 6, lines 53-60, FIG. 2, item 220, and related text));

means for automatically identifying potential software problems from rules set analysis result (see for example column 6, lines 61-65); and,

means for reporting said problems to enable correction of instances where one or more of best practices violations and common coding errors may occur (see for example FIG. 2, item 205, and related text).

Although Saxe discloses a method for automatically applying a set of rules to said program flow analysis graphs (see for example column 6, lines 53-60, FIG. 2, and related text), Saxe does not disclose a static analysis framework comprising: means for automatically generating program graphs, including at least performing a reachability analysis for at least removing one or more edges to reduce reachability.

However, Chan in the same analogous art of program optimization discloses astatic analysis framework comprising: means for automatically generating program graphs, including at least performing a reachability analysis for at least removing one or

more edges to reduce reachability (see for example, FIG. 4, blocks 404, 408, and associated text, e.g. page 4, [0057], lines 1-15).

Therefore, at the time the invention was made it would have been obvious to a person of ordinary skill in the art to modify the teachings of Saxe to include a method as taught above in Chan so as to optimize the program since fewer resources are used in memory (see for example Chan, page 4, [0057]).

**Claim 21:**

Saxe further discloses the static analysis framework as claimed in Claim 20, wherein said rules set represents one or more selected from the group comprising: use of best practices and common coding errors, or combinations thereof (see for example column 7, lines 27-45).

**Claim 22:**

Saxe further discloses the static analysis framework as claimed in Claim 20, wherein said software code comprises scalable componentized applications according to a software development platform (see for example FIG. 1, item 120, FIG. 2, item 220, and related text).

**Claim 23:**

Saxe further discloses the static analysis framework as claimed in Claim 18, wherein said program graphs include one or more selected from the group comprising: a control analysis graph, a data flow analysis graph (see for example FIGs. 7 & 8, and related text), an intraprocedural control flow graph and an inteprocedural control flow graph, said static analysis technique automatically generating a respective one of said control

analysis graph, data flow analysis graph, intraprocedural control flow graph and inteprocedural control flow graph from said software code (see for example column 6, lines 38-44).

**Claim 24:**

Saxe further discloses the static analysis framework as claimed in Claim 23, further including means for automatically identifying classes, fields, methods and class attributes, said set of rules being further applied to said classes and class attributes (see for example FIG. 2, item 202, and related text).

**Claim 25:**

Saxe further discloses the static analysis framework as claimed in Claim 23, wherein said static code analysis further includes automatically identifying attributes of classes, methods, fields, and aspects of a program's body (see for example FIG. 2, item 202, and related text).

**Claim 26:**

Saxe further discloses the static analysis framework as claimed in Claim 20, wherein said means for automatically generating program graphs includes means for performing graph rewriting (see for example FIG. 13, step 1304, and related text).

**Claim 27:**

Saxe further discloses the static analysis framework as claimed in Claim 26, wherein results of said graph rewriting include a run-time characteristics model for said program (see for example column 7, lines 27-37).

**Claim 28:**

Saxe further discloses the static analysis framework as claimed in Claim 26, wherein said means for automatically generating program graphs includes: means for performing a teachability analysis, said reachability analysis being performed with or without constraints (see for example column 11, lines 10-17).

**Claim 29:**

Saxe further discloses the static analysis framework as claimed in Claim 28, wherein said rule search engine automatically applies a set of rules to said rewrite graph results, reachability analysis results and attributes to identify one or more of: possible performance errors or problems concerning correctness, security and privacy of said software code (see for example column 6, lines 5-21).

**Claim 30:**

Claim 30 is a computer program device readable by a machine version of the claimed method step discussed in claim above, wherein all claim limitations have been addressed and/or covered in cited areas as set forth above. Thus, accordingly, these claims are also anticipated by Saxe and Chan.

***Conclusion***

6. Applicant's amendment necessitated the new ground(s) of rejection presented in this Office action. Accordingly, **THIS ACTION IS MADE FINAL**. See MPEP § 706.07(a). Applicant is reminded of the extension of time policy as set forth in 37 CFR 1.136(a).

A shortened statutory period for reply to this final action is set to expire THREE MONTHS from the mailing date of this action. In the event a first reply is filed within TWO MONTHS of the mailing date of this final action and the advisory action is not mailed until after the end of the THREE-MONTH shortened statutory period, then the shortened statutory period will expire on the date the advisory action is mailed, and any extension fee pursuant to 37 CFR 1.136(a) will be calculated from the mailing date of the advisory action. In no event, however, will the statutory period for reply expire later than SIX MONTHS from the date of this final action.

Any inquiry of a general nature of relating to the status of this application or proceeding should be directed tot eh TC 2100 Group receptionist whose telephone number is (571) 272 2100.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll free).

AYC

TUAN DAM
SUPERVISORY PATENT EXAMINER